DOI: 10.15514/ISPRAS-2023-35(6)-5

Язык программирования для обучения технологиям компиляции и трансформации

А.Е. Недоря, ORCID: 0000-0001-8998-7072 <aleksei.nedoria@yandex.ru>

Аннотация. Обучение студентов технологиям компиляции и трансформации является актуальным. В статье рассматриваются критерии выбора языка программирования для выполнения практических работ в качестве целевого, дается краткое описание языка Тривиль и рассматривается пригодность этого языка для использования в соответствии с критериями.

Ключевые слова: языки программирования; обучение технологии компиляции; разработка языков программирования; компиляторы; технологии компиляции.

Для цитирования: Недоря А.Е. Язык программирования для обучения технологиям компиляции и трансформации. Труды ИСП РАН, том 35, вып. 6, 2023 г., стр. 95–102. DOI: 10.15514/ISPRAS–2023–35(6)–5.

Programming Language for Teaching Compilation and Transformation Technologies

A.E. Nedoria, ORCID: 0000-0001-8998-7072 <aleksei.nedoria@yandex.ru>

Abstract. Teaching students compilation and transformation technologies is in demand. The article discusses the criteria for choosing a programming language for practical work as a target language, gives a brief description of the Trivil language and examines the suitability of this language for using in accordance with these criteria.

Keywords: programming languages, language design; compilers; compilation technologies.

For citation: Nedoria A.E. Programming language for teaching compilation and transformation technologies. *Trudy ISP RAN/Proc. ISP RAS*, vol. 35, issue 6, 2023. pp. 95-102 (in Russian). DOI: 10.15514/ISPRAS-2023-35(6)-5.

1. Введение

Разработка языков программирования, компиляторов и инструментов разработки, не теряет своей актуальности, а скорее даже становится более актуальной, что обусловлено как минимум двумя факторами

1. лицензионные проблемы, связанные с глобальным политическим противостоянием;
2. и появление новых идей и новых требований, воплощенных, например, в языках Rust [1], Go [2], Swift [3], Mojo [4].

Соответственно, является актуальным обучение студентов технологиям компиляции и трансформации программ и технологиям разработки или расширения языков программирования.

Обучение технологиям должно, наряду с теорией, включать практические работы, например:

- разработка компилятора, или, скорее, отдельных проходов компилятора, включая анализ, оптимизации, генерацию кода;
- тестирование компилятора, включая проверку соответствия компилятора языку (compliance test);
- расширение языка, добавление новых конструкций языка;
- разработка среды исполнения, включая управление памятью и сборку мусора;
- разработка библиотек;
- разработка инструментов, к примеру, статического анализатора;
- проведение доказательства свойств языка, например, type system soundness;
- построение модели памяти или параллельный вычислений;
- и так далее.

Для каждой из таких работ нужен язык программирования, который используется, как полигон для студенческих работ. То есть для этого языка пишется или дорабатывается компилятор, разрабатывается среда выполнения, проводятся доказательства и так далее. Будем называть такой язык **целевым** языком.

2. Сложности выбора целевого языка

Очевидно, что выбор такого языка существенно влияет на качество обучения. При выборе существенным фактором является ограниченность времени (семестр, год), за который студент должен показать работающий результат. Практическая работа должна быть, с одной стороны, достаточно серьезной, с другой стороны она существенно ограничена по времени выполнения.

Какой язык может быть выбран в качестве целевого?

По сути, есть три варианта:

- существующий язык программирования;
- подмножество существующего языка;
- учебный (игрушечный) язык.

Замечу, что в этой статье я, во многом, опираюсь на свой студенческий опыт. Моя дипломная работа в НГУ в 1984 году называлась «Компилятор с языка Эдисон для МВК Эльбрус». Компилятор был написан на языке Эль-76 [5] группой из трех студентов в течение года. В качестве целевого языка был выбран язык Эдисон (Edison) [6], автором которого был Пер Бринч Хансен. Эта дипломная работа во многом определила мой профессиональный путь, позволив мне накопить опыт как в проектировании и разработке достаточно сложной программы, так и в управлении командой разработчиков.

В то время я не задумывался, почему руководитель моей дипломной работы выбрал именно этот язык, но сейчас очевидны три существенных достоинства:

- простота языка
- хорошая (и короткая, около 35 страниц) спецификация языка
- современность языка: описание языка появилось в 1981 году

Вернусь к выбору целевого языка программирования. Среди широко используемых языков, ни один из языков нельзя назвать простым. И даже, если мы не говорим о C++, и судим только по размеру спецификации языка, то спецификация языка Kotlin - более 300 страниц, Java – 800 страниц, Go – 107 страниц. Можно сказать, что Go относительно простой язык, но простым его назвать нельзя.

Для работы с любым таким языком студенту понадобится существенное время на изучение самого языка. При этом даже если студент до этого писал на этом языке, то этих знаний недостаточно, любая практическая работа из тех, что перечислены выше, потребует более глубокого понимания языка, а работы, связанные с разработкой или доработкой компилятора (или других инструментов), еще и знаний устройства компилятора и инструментов.

Остальные два варианта — выделение подмножества существующего языка или создание учебного языка представляются возможными, но очень трудоемкими для преподавателя. Преподавателю, кроме разработки подмножества или языка, придется написать компилятор или обрезать компилятор (для подмножества) и выполнить другие предварительные работы.

В этой работе, я предлагаю язык, который можно отнести как к 1-му варианту, так и к 3-му – это использование языка Тривиль [7], который разрабатывался для решения других задач, но является (как и Эдисон в свое время) простым, современным и описанным в короткой спецификации.

Замечу, что я не предлагаю использовать Тривиль, как язык для обучения программирования или как **инструментальный** язык, на котором студенты решают свою практическую задачу. Использование целевого языка в качестве инструментального возможно, для некоторых задач это может быть полезно, но не для всех.

3. Критерии оценки целевого языка

Прежде чем переходить к краткому описанию языка Тривиль, выделим критерии применимости некоторого языка (или подмножества) для использования в качестве целевого, чтобы потом оценить язык Тривиль по этим критериям.

На мой взгляд, важнейшим критерием является обозримость и достаточная простота целевого языка, что позволяет студенту:

- видеть свою задачу целиком;
- сосредоточится на своей задаче (а не на сложностях самого язык).

Думаю, что это очевидно, если рассматривать, например, практическую задачу разработки части компилятора или доказательство type system soundness.

В то же время, язык не должен быть слишком простым, чтобы обучение было эффективным.

Другими важными критериями являются:

- наличие спецификации языка разумного размера;
- современность синтаксиса и семантики языка;
- наличие компилятора на популярных платформах;
- простота и обозримость компилятора и инструментов;
- открытый код и отсутствие лицензионных ограничений.

В табл. 1 перечислены критерии для целевого языка в таблице со степенью важности критерия (очень важно, важно, полезно, не важно), и для сравнения рассмотрим степень важности для инструментального языка.

4. Язык программирования Тривиль

Работа над языком Тривиль была начата в ноябре 2022 года с целью получить язык и инструментальные средства минимально достаточные для удобной разработки нескольких компиляторов разрабатываемого семейства языков. К лету 2023 года было сделано (все исходные тексты в открытом репозитории [8]):

- описание языка [7],
- два компилятора - первый на Go, второй на Тривиле

- и набор стандартных библиотек.

Оба компилятора построены по классической архитектуре:

- Парсер строит AST
- Далее по AST проходит семантический анализ (два прохода)
- По корректному атрибутированному AST делается генерация в C99

Табл. 1. Критерии целевого и инструментального языков
Table 1. Criteria for the applicability of target and instrumental languages

Критерий	**Для целевого языка**	**Для инструментального языка**
Обозримость и простота языка	Очень важно	Полезно
Описание (спецификация) языка разумного размера	Очень важно	Полезно
Современность синтаксиса и семантики языка	Важно	Полезно
Открытый код компилятора и экосистемы, отсутствие лицензионных ограничений	Очень важно	Полезно
Простота (обозримость) компилятора и инструментов	Очень важно (но не для всех практических работ)	Не важно
Доступность на популярных платформах, удобство использования	Важно	Очень важно

Размеры кода:

- компилятор на Go: 11,200 строк
- компилятор на Тривиле: 10,500 строк
- runtime на C99: 1,800 строк
- библиотеки на Тривиле: 2,800 строк

Первый доклад об языке был сделан на семинаре STEP-2023 [9] в апреле 2023. При подготовке к семинару пришло понимание, что язык может быть использован в качестве целевого для обучения студентов.

Приведем несколько примеров, которые должны быть понятны без пояснений или с минимальными пояснениями. Начнем, традиционно, с программы “Привет, мир” (см. Листинг 1).

```
модуль привет
импорт "стд::вывод"
вход{
   вывод.ф("Привет, мир\n")
}
```

Листинг 1. Привет, мир
Listing 1. Hello, world

Немного более сложный пример, показывающий описание функций показан на Листинге 2, результат исполнения примера: `5! = 120`.

```
модуль факториал
импорт "стд::вывод"
фн Факториал(n: Цел64): Цел64 {
    надо n > 1 иначе вернуть 1
    вернуть n * Факториал(n - 1)
```

```
}
вход{
    пусть № = 5
    вывод.ф("$;! = $;\n",  №, Факториал(№))
}
```

Листинг 2. Факториал
Listing 2. Factorial

В рамках статьи сделать подробное описание языка не представляется возможным (см. [7]), в табл. 2 перечислены его основные черты.

Табл. 2. Основные черты языка
Table 2. The main features of the language

Типы	**Описания**

• Байт, Цел64, Слово64
• Вещ64
• Лог
• Символ (unicode)
• Строка, Строка8
• тип вектора: []T
• тип класса: **класс** (база) {}
• может быть тип: **мб** T
	• **тип** T = тип
	• **конст** к (: T)? = знач
	  ○ **конст** к = 1
	  ○ **конст** к: Байт = 1
	• **пусть** п(: T)? (= | :=) знач
	  ○ **пусть** № = 1 // val
	  ○ **пусть** №: Байт := 1 // var
	• описания функций и методов

Описание функций
`фн Факториал(n: Цел64): Цел64 {`
`  // операторы`
`}`
	Описание методов
	`фн (сб: Сборщик) добавить строку(ст: Строка){`
	`  // операторы`
	`}`

Операторы	**Выражения**
• :=, ++, -- • **если** усл {} **иначе** {} • **надо** усл **иначе** (завер | {}) • **выбор** выражение {} • **выбор** тип перем = выр {} • **пока** усл {} • **цикл** перем **среди** век {} • **прервать** • **вернуть** знач? • **авария**(“описание”)	• +, -, *, /, % • =, #, <, <=, >, >= • логические: &, |, ~ (not) • битовые: :&, :|, :\ (xor), :~, <<, >> • преобразование типа: (: • подтверждение типа: ^ • конструктор вектора: T[1, 2, 3] • конструктор класса: T{имя: "Вася" }

При разработке языка не ставилась задача придумать новые конструкции языка, скорее, как и предполагает имя языка, ставилась задача использовать проверенные конструкции. Тем не менее, в языке есть интересные особенности, обеспечивающие удобство и безопасность программирования:

- Лаконичный синтаксис
- Обязательная инициализация полей и переменных
- Изменяемые и неизменяемые поля и переменные
- Поздняя инициализация для полей
- Простой вывод типов (type inference)
- Создание экземпляров классов и векторов с помощью конструкторов (Go composite literals)
- Отсутствие явных ссылок (ссылочные типы - класс, вектор, Строка)
- Безопасная работа со ссылками (null safety)
- Операторы “надо” и “выбор” по типу

- Безопасные вариативные и полиморфные параметры
- Обобщенные модули
- Русскоязычный синтаксис и ключевые слова, пробелы в идентификаторах

Часть из этих особенностей показана в следующих примерах.

Пример: реализации класса «Сборщик» (string builder), текст взят из библиотеки «стд::строки» (см. Листинг 3).

```
модуль строки

тип Байты = []Байт // тип: вектор байтов
тип Сборщик* = класс { // экспортированный класс Сборщик
    байты = Байты[]  // конструктор вектора длины 0
    число-символов := 0
}
фн (сб: Сборщик) добавить строку*(ст: Строка) {/*...*/ }
фн (сб: Сборщик) строка*(): Строка {
    вернуть сб.байты(:Строка) // преобразование к UTF-8 строке
}
```

Листинг 3. Класс «Сборщик»
Listing 3. Class "StringBuilder"

Второй пример из библиотеки «строки» показывает реализацию форматного вывода, класс «Разборщик» (см. Листинг 4).

```
модуль  строки
тип Символы = []Символ // вектор Unicode символов
тип Разборщик = класс {
    сб: Сборщик = позже // поле с поздней инициализацией
    формат: Символы = позже
}
// Добавляет строку по формату
фн (сб: Сборщик) ф*(фс: Строка, аргументы: …*) {
    // конструктор экземпляра класса с инициализацией полей:
    пусть р = Разборщик{сб: сб, формат: фс(:Символы)}
    пока р.следующий() {
        надо №-арг < длина(аргументы)
        иначе авария("не достаточно аргументов")
        // здесь обработка аргумента
    }
}
```

Листинг 4. Класс «Разборщик»
Listing 4. "Parser"

Следующий пример показывает работу с может-быть типами (безопасными ссылками, null safety) на примере бинарного дерева (см. Листинг 5). Ссылки на под-узлы бинарного дерева описаны с помощью *может быть* типа, которые добавляет значение «пусто» к домену значений типа Узел. Функция «число узлов» выдает число узлов бинарного дерева. Операция `"^"` является аналогом операции `"!"` (non-null assertion) в таких языках, как Kotlin [10] и Typescript.

Надеюсь, что примеры дает некоторое общее понимание языка, подробнее, см. [8].

```
тип Узел = класс {
    № := 0
    лев: мб Узел := пусто
```

```
    прав: мб Узел := пусто
}
фн число узлов(у: мб Узел): Цел64 {
    если у = пусто { вернуть 0 }
    вернуть 1 + число узлов(у^.лев) +  число узлов(у^.прав)
}
```

Листинг 5. Бинарное дерево
Listing 5. Binary Tree

5. Оценка использования в качестве целевого языка

Дадим оценку языку Тривиль как целевому языку в соответствии с критериями определенными выше (см. табл. 3).

Табл. 3. Применимость языка Тривиль в качестве целевого языка
Table 3. Trivil's applicability as a target language

Критерий	**Для целевого языка**	**Тривиль**
Обозримость и простота языка	Очень важно	Да
Описание (спецификация) языка разумного размера	Очень важно	Да размер описания языка — 40 страниц
Современность синтаксиса и семантики языка	Важно	Да
Открытый код компилятора и экосистемы, отсутствие лицензионных ограничений	Очень важно	Да
Простота (обозримость) компилятора и инструментов	Очень важно	Да
Доступность на популярных платформах, удобство использования	Важно	Платформы: Windows, Linux, FreeBSD, MacOS IDE: слабая интеграция

Единственным пунктом, в котором Тривиль не полностью удовлетворяет критерию, это «удобство использования». В настоящее время полноценная интеграция языка в IDE отсутствует. Впрочем, такая интеграция может быть темой практической работы студентов.

Было бы интересно сравнить Тривиль по этим критериям с другими языками, используемыми в качестве целевых, например, с языком РуСи [11], авторы которого ставят, в качестве одной из задач: «облегчить изучение программирования школьниками и студентами». К сожалению, такой информации у меня нет. Сравнение с общеизвестными языками не имеет смысла, если нет информации об их использовании в качестве целевых.

6. Заключение

За год с начала разработки языка Тривиль, язык прошел практическую проверку — на нем был написан компилятор и набор библиотек. По ходу разработки, в язык было внесено несколько существенных улучшений. Кроме того, язык два раза обсуждался на семинаре STEP-2023, обратная связь от участников семинара была очень полезна.

Начиная с сентября 2023 года, язык проходит практическую проверку в качестве целевого языка. В Университете Иннополиса несколько студенческих команд используют Тривиль и его компиляторы в практических работах по генерации кода для нескольких платформ и по созданию теста на соответствие компилятора и языка (language compliance test suit).

На взгляд автора, Тривиль может стать основой полигона для обучения студентов в разных вузах. Под полигоном я понимаю набор языков, компиляторов, других инструментов, формальных моделей и доказательств, которые используются студентами и куда выкладываются результаты практических работ.

Это позволит использовать результаты в следующих работах, наращивая разнообразие задач, в том числе это позволит вносить соревновательный элемент, например, лучшие реализация, инструмент или доказательство. Для этого нужны те, кто заинтересован в повышении качества обучения и место, где собирается информация о практических работах и их результатах.

Список литературы / References

[1]. Klabnik S., Nichols C.. The Rust Programming Language. Available at: https://doc.rust-lang.org/book/title-page.html, accessed 29.10.2023.
[2]. The Go Programming Language Specification. Available at: https://golang.org/ref/spec, Version of Aug 2, 2023.
[3]. The Swift Programming Language. Available at: https://docs.swift.org/swift-book/documentation/the-swift-programming-language/aboutthelanguagereference/, accessed 29.10.2023.
[4]. Mojo Programming Manual. Available at: https://docs.modular.com/mojo/programming-manual.html, accessed 29.10.2023.
[5]. Пентковский В. М. Автокод Эльбрус. Принципы построения языка и руководство к пользованию. М., Наука, 1982, 352 с.
[6]. Brinch Hansen, Per. The Design of Edison. Software: Practice and Experience Vol. 11, No. 4, 1981, pp 363-396. DOI: 10.1002/SPE.4380110404.
[7]. Язык программирования Тривиль, версия языка 0.9.2. Available at: https://gitflic.ru/project/alekseinedoria/trivil-0/blob?file=doc%2Freport%2Freport.pdf, accessed 29.10.2023.
[8]. Репозиторий язык программирования Тривиль и компиляторов. Available at: https://gitflic.ru/project/alekseinedoria/trivil-0, accessed 29.10.2023.
[9]. Российский гибридный семинар STEP-2023 по фундаментальным вопросам программной инженерии, теории и экспериментальному программированию. Available at: https://persons.iis.nsk.su/en/STEP-2023, accessed 29.10.2023.
[10]. Null Safety. Available at: https://kotlinlang.org/docs/null-safety.html, accessed 29.10.2023.
[11]. Терехов А. Н,, Терехов М. А.. Проект РуСи для обучения и создания высоконадежных программных систем. Available at: https://cyberleninka.ru/article/n/proekt-rusi-dlya-obucheniya-i-sozdaniya-vysokonadezhnyh-programmnyh-sistem, accessed 29.10.2023.

Информация об авторах / Information about authors

Алексей Евгеньевич НЕДОРЯ, к.ф.-м.н. Сфера научных интересов: языки программирования, программные экосистемы, технологии программирования, мульти-платформенное программирование, архитектурной программирования, компонентно-ориентированное программирование.

Aleksei Evgenevitch NEDORIA – Cand. Sci. (Phys.-Math.). Research interests: programming languages, software ecosystems, programming technologies, multi-platform programming, architecture programming, component-oriented programming.